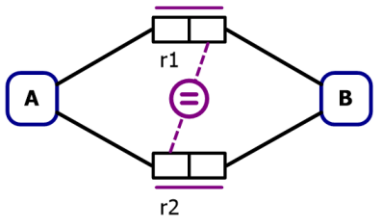
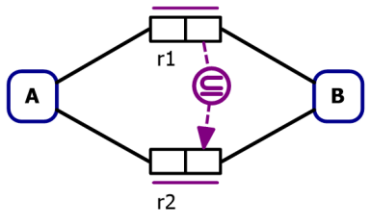
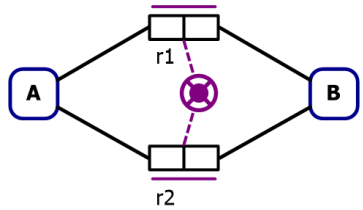
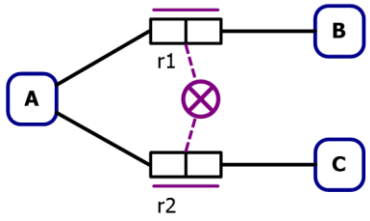
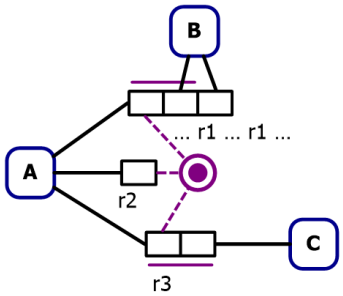
Sensorveiledning for IN2090 og INF1300

6. desember 2018 14:30 – 18:30 (4 timer)

1. Eksterne skranker (5%)

I modellene nedenfor (ORM2) skal du anta at alle begreper har en unik representasjon.

Er plasseringen av den eksterne skranken gyldig eller ugyldig? *Automatisk vurdert*

	**Ugyldig** ❌
	**Gyldig** ✔
	**Ugyldig** ❌
	**Ugyldig** ❌
	**Gyldig** ✔

2. Modellering i ORM (maks 35 poeng)

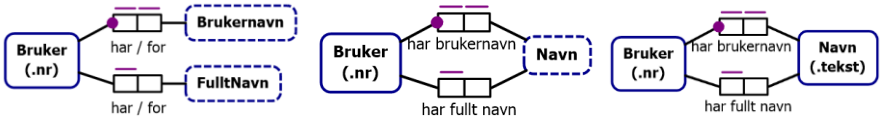
Det viktigste er at studentene lager hensiktsmessige begreper og verdityper, og at rollene mellom dem svarer på oppgaveteksten. Pass også på påkrevde roller og at entydighetsskrankene svarer på oppgaveteksten (de fleste er angitt ganske eksplisitt i oppgaveteksten). I alle faktatyper og broer skal det være rollenavn (rollebeskrivelser). Det holder at det er rollenavn som kan leses én vei.

Veiledende poengfordeling:

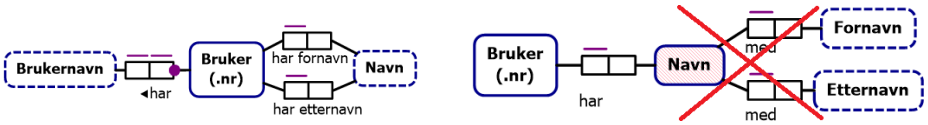
- **1a, 1b, 1c**: Maks 4. **1d**: Maks 6. **1e**: Maks 2.
- Hver oppgave i del 2 skal telle like mye (maks 5 poeng for hver av 2a, 2b og 2c)

Generelt bør det trekkes 1 poeng **per** manglende/feil interne entydighetsskranker og påkrevde roller. Mer spesifikt må følgende være på plass i hver deloppgave:

1a: Påkrevd rolle mot brukernavn. Riktige interne entydighetsskranker. Verditypen «FulltNavn» kan kalles «Navn», og kan i så fall også gjenbrukes for brukernavn og emnenavn (med passende rollenavn). Navn kan modelleres som begrep, men må i så fall ha en identifikator (tredje illustrasjon nedenfor).



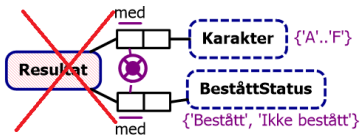
Om fullt navn er delt opp i fornavn og etternavn, må dette være gjort hensiktsmessig (dvs ingen eksterne skranker, ugyldige ekstrabegrep e.l.).



1b: Påkrevd rolle mot emnenavn. Riktige interne entydighetsskranker.

1c: Én *binær* faktatype med riktig entydighetsskranke. -2 poeng for manglende exclusion-skranke ⊗. Trekk -1 for *feil plassering* av skranken (mellom feil roller).

1d: Bør modelleres som faktatype med 4 roller. *Spesielt viktig* med riktig entydighetsskranke, -1 poeng ved gyldig skranke på feil roller, -2 om skranken er ugyldig (over 1-2 roller) eller mangler. Resultat kan være et begrep, eller ResultatVerdi-verditype. Verdiskranke på resultatobjektet (-1 ved mangel; syntaks er ikke viktig). -2 poeng om Resultat er et begrep med to mulige verdityper, men mangler identifikator.



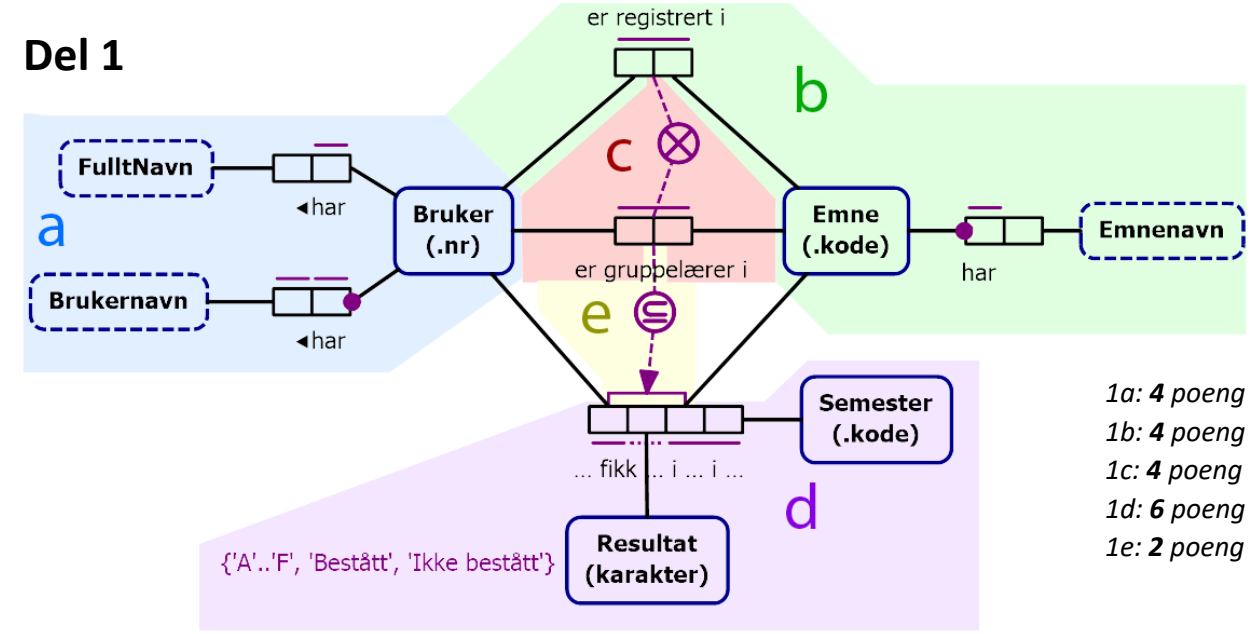
1e: -1 poeng ved feil plassering (mellom feil roller). -1 poeng om retningen er feil (eller mangler). 0 poeng for feil skranke. Hvordan pilen(e) er tegnet er uvesentlig, så lenge det er tydelig hvilke roller som pekes på.

2a: Spesielt ekstern entydighetsskranke er viktig. Påkrevde roller. -1 poeng per feil skranke/påkrevd rolle. -2 ved manglende oblignr, eller hvis nummer er satt som eneste identifikator.

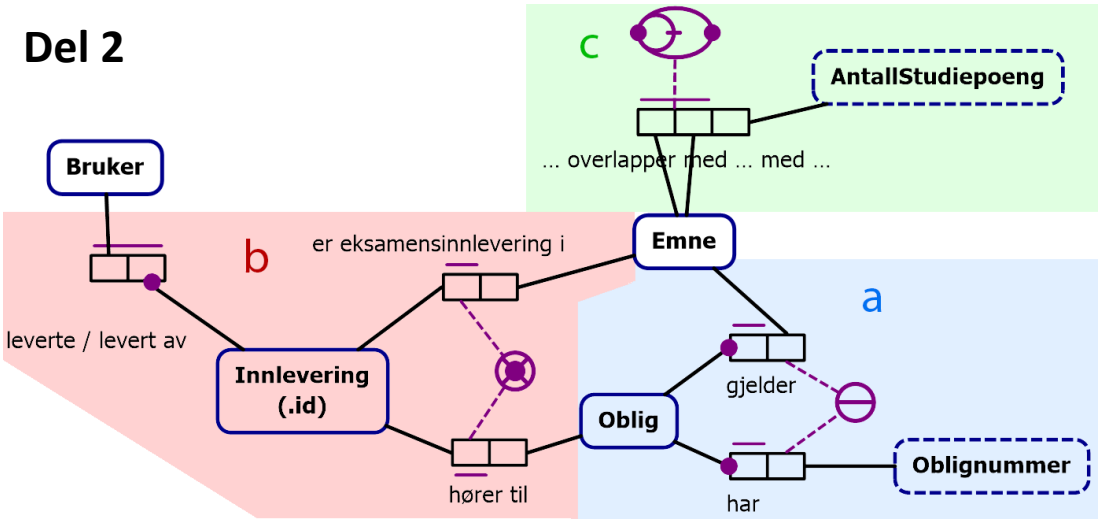
2b: Faktatype mot oblig og emne. Plassering av exclusive-or-skranke (-1 for feil plassering, -1 for feil skranke, -2 for hvis xor-skranke mangler). -2 poeng for manglende identifikator på Innlevering eller identifikator som ikke gir mening (f.eks. ekstern entydighetsskranke).

2c: Ternær faktatype (-2 poeng ved binær faktatype uten verdi/begrep for poeng). Riktig entydighetsskranke er ekstra viktig her (-2 ved feil). -1 ved feil *plassering* av ringskranke. Symmetrisk + irrefleksiv ringskranke kan tegnes som én skranke (akkurat denne kombinasjonen står også på side 5 i vedlegget), men å tegne ringskrankene hver for seg er også greit. -1 *per* manglende ringskranke.

Fullt løsningsforslag (de ulike deloppgavene er merket med ulike farger):



*1a: **4** poeng*
*1b: **4** poeng*
*1c: **4** poeng*
*1d: **6** poeng*
*1e: **2** poeng*



*2a: **5** poeng*
*2b: **5** poeng*
*2c: **5** poeng*

3. Realisering (maks 10 poeng)

Himmellegeme(navn, [oppdaget_årstall], [i_bane_rundt])
Romfartøy(navn, oppskytningsdato, status)
Besøk(himmellegeme, romfartøy, dato, oppdragstype)

Fremmednøkler:

- Himmellegeme(i_bane_rundt) → Himmellegeme(navn)
- Besøk(himmellegeme) → Himmellegeme(navn)
- Besøk(romfartøy) → Romfartøy(navn)

Realiseringen gir også relasjonene Dato(ymd) og År(tall). Disse relasjonene kan undertrykkes (slettes), og er derfor valgfrie å ha med. Studenten må selv finne et passende navn for relasjonen Besøk. Navnene på attributtene kan være annerledes, gitt at de i tilstrekkelig grad er beskrivende. Merk at oppgaven også ble gitt på engelsk/nynorsk med ulike språk i modellene (men de er ellers like).

Her er det mange mulige feil som kan oppstå. I utgangspunktet bør det trekkes ca 1 poeng per feil (eller manglende) attributt/tabell/fremmednøkkel.

SQL-oppgaver

Det er greit å løse deler av oppgavene ved å lage et view som så brukes «hovedspørringen». Alternativ syntaks som implisitt join og JOIN uten ordet INNER er tillatt. Bruk av alias er valgfritt. Store/små bokstaver i syntaksen, plassering av whitespace osv. skal ikke slå ut på poenggivning.

Der besvarelsene avviker sterkt fra løsningsforslaget brukes skjønn.

4. SQL, create (maks 5 poeng)

Skriv en CREATE-spørring for tabellen orders med passende datatyper for attributtene, slik at tabellen blir som beskrevet, inkludert primær- og fremmednøkler.

- ***-1 poeng** for manglende/ugyldig definert primærnøkkel (f.eks. UNIQUE)*
- ***-1 poeng** for manglende/ugyldig definert fremmednøkkel. Manglende kolonnenavn i «`REFERENCES customers`» er tillatt.*
- ***-1 per ugyldige datatype** (f.eks. til sammen -1 om type STR brukes begge stedene i stedet for VARCHAR, men -2 om man både bruker STR for VARCHAR og NUMBER for INT).*
  CHAR(n) og VARCHAR(n) er tillatt hvis en hensiktsmessig lengde er brukt. DATE er ikke tillatt.
- ***Trekk for feil syntaks** som er mer enn en «glipp», som gjennomgående manglende komma, typer før kolonnenavn osv. **-1** per slike feil.*
- *Ingen trekk for manglende NOT NULL. CHECK for status gir verken trekk eller plusspoeng.*

```sql
CREATE TABLE orders (
    id INT PRIMARY KEY, -- eller INTEGER
    customer_id INT NOT NULL REFERENCES customers(id),
    shipping_addr VARCHAR NOT NULL,
    time_entered TIMESTAMP NOT NULL,
    status VARCHAR NOT NULL -- eller f.eks. TEXT, CHARACTER VARYING, VARCHAR(n), CHAR(n)
    -- PRIMARY KEY (id) -- også tillatt
    -- FOREIGN KEY (customer_id) REFERENCES customers(id) -- også tillatt
);
```

5. SQL, view (maks 5 poeng)

Definer et view delayed_orders(id, customer_id, time_entered) som inneholder alle ordrer som har status 'delayed'.

3 poeng *for riktig opprettelse av viewet med matchende kolonner i SELECT-delen*
2 poeng *for riktig FROM- og WHERE-del*

```
CREATE VIEW delayed_orders AS
-- eller
CREATE VIEW delayed_orders(id, customer_id, time_entered) AS
    SELECT id, customer_id, time_entered
    FROM orders
    WHERE status = 'delayed';
```

6. SQL, kunder og ordrer (maks 5 poeng)

Skriv en spørring som returnerer alle kunder der navnet begynner med «Ola», samt deres ordrer hvis de har noen. Ta også med kunder som ikke har kjøpt noe. Resultatet skal inneholde kundens navn og adresse, samt ordrenes id og status, sortert etter registreringstidspunkt slik at de siste registrerte kundene kommer først.

2 poeng *for bruk av `LEFT [OUTER] JOIN` (eller tilsvarende)*
1 poeng *for LIKE og % i WHERE (eller tilsvarende)*
1 poeng *for riktig sortering (poenget trekkes i sin helhet om sorteringen ikke er gjort helt riktig)*

```
SELECT name, address, o.id, o.status
FROM customers c
  LEFT JOIN orders o ON c.id = o.customer_id
WHERE name LIKE 'Ola%'
ORDER BY c.time_registered DESC;
```

7. SQL, aggregering (maks 5 poeng)

Skriv en spørring som finner antall ordrer per kunde for hver forskjellig status. Skriv ut kundens id og navn, samt antall og status.

4 poeng *for korrekt bruk av COUNT og GROUP BY (men ok om count brukes med kolonnenavn)*
1 poeng *som kan trekkes i sin helhet for andre feil i SELECT eller FROM-delen*

```
SELECT c.id, c.name, COUNT(*), o.status
FROM customers c
  INNER JOIN orders o ON c.id = o.customer_id
GROUP BY c.id, c.name, o.status;
```

8. SQL, kunder med flere ordrer (maks 5 poeng)

Skriv en spørring som finner alle kunder som har minst to ordrer med forskjellig shipping_address. Skriv ut kundenes id og navn.

1 poeng *for riktige kolonner og join mellom customers og orders i FROM-delen.*
4 poeng *for riktig logikk for å hente ut kunder som har ordre med ulik shipping_address*

```sql
-- med self-join
SELECT DISTINCT c.id, c.name
FROM customers c
  INNER JOIN orders o1
    ON c.id = o1.customer_id
  INNER JOIN orders o2
    ON o1.customer_id = o2.customer_id AND o1.shipping_addr != o2.shipping_addr;

-- med gruppering og HAVING
SELECT c.id, c.name
FROM customers c
  INNER JOIN orders o ON c.id = o.customer_id
GROUP BY c.id, c.name
HAVING COUNT(DISTINCT shipping_addr) > 1; -- 1 poeng trekk ved manglende DISTINCT
```

9. SQL, produkter bestilt oftest (maks 10 poeng)

Skriv en spørring som returnerer alle kunder som har bestilt de produktene som har vært bestilt oftest (altså de produktene som finnes på flest forskjellige ordrer). Skriv ut kundenes id.

Her er det spesielt viktig at studentene identifiserer at det er behov for subspørringer, og at disse er brukt på en korrekt måte.
4 poeng *for å finne antall ordre et produkt finnes på*
3 poeng *for å riktig finne det høyeste antallet produkter*
1 poeng *for å unngå duplikate kunde-ID-er (f.eks. med DISTINCT)*
2 poeng *for resten (joine riktige tabeller, hente customer_id o.l.)*

```sql
WITH num_sold AS (
  SELECT product_id, COUNT(order_id) as num
  FROM order_items
  GROUP BY product_id
)
SELECT DISTINCT o.customer_id
FROM num_sold
  INNER JOIN order_items oi ON num_sold.product_id = oi.product_id
  INNER JOIN orders o ON oi.order_id = o.id
WHERE num_sold.num = (SELECT MAX(ns.num) FROM num_sold ns);
```

10. SQL, total kostnad (maks 10 poeng)

Skriv en spørring som finner antall enheter og total kostnad for alle produkter solgt til kunder i USA, hvor minst 10 enheter har blitt solgt. Skriv ut produktenes id og navn, samt antall enheter og total kostnad.

Her bør alle delene vektlegges likt:
- Riktig gruppering
- avgrensning med having eller tilsvarende
- join med 4 tabeller og filtrering på country = 'USA'
- Riktig seleksjon med utregning av antall og kostnad.

```
SELECT
  oi.product_id,
  p.name,
  SUM(quantity) AS antall,
  SUM(quantity*unit_price) AS kostnad
FROM order_items oi
  INNER JOIN orders o ON oi.order_id = o.id
  INNER JOIN customers c ON o.customer_id = c.id
  INNER JOIN products p ON oi.product_id = p.id
WHERE c.country = 'USA'
GROUP BY oi.product_id, p.name
HAVING SUM(quantity) >= 10;
```

11. Relasjonsteori (maks 5 poeng)

a. Hvilke attributter i tabellen *customers* er kandidatnøkler? ***Maks 1 poeng***

id og {pnr, fdate}

b. Hvilken normalform er tabellen *customers* på? Begrunn svaret ditt. ***Maks 4 poeng***

Vi har følgende ikke-trivielle FD-er:
- id → pnr, fdate, name, address, country, time_registered
- {pnr, fdate} → id, name, address, country, time_registered

I begge FD-ene er venstre side en supernøkkel. Dermed er begge FD-ene på BCNF.
Tabellen er derfor på **BCNF**.

Det er også greit å si at venstre side er en kandidatnøkkel.